Final Project

Canonical Pairwise Alignments

Math 343

Jacky Ken Tse

301301625

Combinatorial Specification for Canonical Pairwise Alignments:

A canonical pairwise alignment can be of the following forms:

- Empty (when n = 0)
- If n > 0 then we have 3 cases,
    - We get a pair of elements, either mismatch or matching, which will contribute a size of 2, and a canonical pairwise alignment with a size of n-2. (Note it can also be empty if n-2 = 0.
    - We get a deletion, which will contribute a size of 1, and a canonical pairwise alignment of size n-1. Can also possibly be an empty canonical pairwise alignment.
    - We get an insertion, which will contribute a size of 1, and a canonical pairwise alignment of size n-1. Can also possible be an empty canonical pairwise alignment.

Using the above we have the following definition for a canonical pairwise alignment.

$$A = Atom$$

$$T = Atom$$

$$G = Atom$$

$$C = Atom$$

$$Alphabet = Union(A, T, G, C)$$

$$Null = Epsilon$$

$$Mat = Prod(Alphabet, Alphabet)$$

$$Del = Prod(Alphabet, Null)$$

$$Insert = Prod(Null, Alphabet)$$

$$W1 = Prod(Mat, Alignment)$$

$$W2 = Prod(Del, Align2)$$

$$W3 = Prod(Insert, Alignment)$$

$$Align2 = Union(Epsilon, W1, W2)$$

$$Alignment = Union(Epsilon, W1, W2, W3)$$

In this specification we have an alphabet, that is the union of the four letter A,T,G,C. Null is being used to represent "-" in the alphabet for a pairwise. Mat is the family for BOTH matches and mismatches in a canonical pairwise alignment. Del is the family for deletions in a canonical pairwise alignment. Lastly Insert is the family for insertion in a canonical pairwise alignment. When we generate an alignment we have 4 possible choices that could happen, either we get an empty alignment, choose W1 where we get a pair of alignment and a canonical pairwise alignment (possibly empty), choose W2 where we get a deletion and a canonical pairwise alignment where we are NOT allowed insertion (or possibly empty alignment), or choose W3 where we ger an insertion and a canonical pairwise alignment. (possibly empty)

W2 ensures that we get only canonical pairwise alignments as this family will ensure that the next object, we get will not be an insertion (since we cannot have deletion followed by an insertion). Notice that W2 calls Align2 which is almost the same as alignment, but we are missing W3 as an object. W3 is an object that has an insertion object followed by an alignment. Align2 will ensure we will not get a delete followed by an insert thus preventing us from double counting. Thus, we can ensure that our specification describes all canonical pairwise alignments and only canonical pairwise alignments.

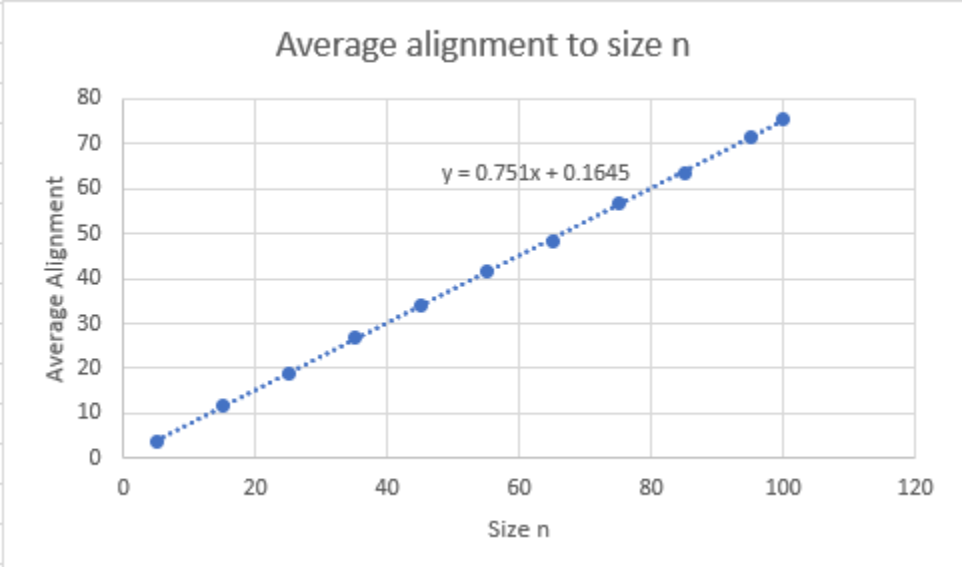
Closed Form Expression of the OGF for Canonical Pairwise Alignments:

The closed form of the OGF for a Canonical Pairwise Alignment is:

$$A(z) = -\frac{1}{8z - 1}$$

Estimates of Matches, Mismatches, Insertions and Deletions:

Using the data for size n from 5 by 5 to 100, and for search size n we generate 100 different alignment we can estimate the expected number of matches, mismatches, deletes and inserts for a given size.



This graph was created in excel using data generated form the maple code using random generation of alignments of size n. The bottom of the graph represents size n where the left hand side shows the average number of alignments needed to from a canonical pairwise alignment of size n. Note when I mean alignment -> (A,G) (-,C) is two alignment for a canonical pairwise alignment of size 3. The best fit line equation:

$$Average\ Alignment = 0.751n + 0.1645$$

For thus the expected number alignments to create a canonical pairwise alignment of size 9 is $0.751 * 9 + 0.1645 = 6.9235$ or roughly 7 alignments. This average alignment data was generated by taking the

total number of alignments needed to generate 100 canonical pairwise alignment of size n and dividing it with 100 (the total number of canonical pairwise alignments).

Taking the averages of proportional matches, mismatches, deletes, and inserts we get the following constants.

Proportional Match	0.080341
Proportional Mismatch	0.240977
Proportional Delete	0.339495
Proportional Insert	0.339969

These values are generated by taking the total number of match or mismatches or deletes or inserts and dividing it with the total number of alignments. These values tell us the approximately how many of the alignments generated for size n are a match, mismatch, delete, or insert. Thus, if there were 100 alignments, 8 of them would be matches, 24 would be mismatches, 34 would be deletes and 34 would be inserts. Also adding them up we would result with 100 total alignments.

Knowing this the expected number of matches, mismatches, deletes, and inserts for size $n \geq 2$ are the following in terms of size n.

$$Matches: 0.080341(0.751n + 0.1645)$$

$$Mismatches: 0.240977(0.751n + 0.1645)$$

$$Deletes: 0.339495(0.751n + 0.1645)$$

$$Inserts: 0.339969(0.751n + 0.1645)$$

The reason this is the estimates for size $n \geq 2$ is that if $n = 1\ or\ 0$ it will not work as intended. (Can't have matches for size 1, similarly can't have deletes for size 0.) Luckily these cases are easy, and we can encapsulate in a piecewise function where for case $n = 0$, we return 0 for all 4 of them which $n = 1$, 0's for match and mismatch while 1 for delete or inserts.

Comments on the Estimates:

Looking purely at the constants (proportional matches, mismatches, deletes, and inserts) we can see that these values are agreeable. Notice how there are only 4 ways to generate a single match for size 2. While there are 12 ways to generate mismatches. This means that mismatches are 3 times more likely to show up as opposed to matches. This reflects our constants as mismatches are roughly 3 times higher than matches. While looking at this analytically, we can easily come to the conclusions that mismatches are 3 times are more likely.

Applying analytical thinking looking at deletes and inserts we would struggle a lot to generate an accurate estimate without the use of random generation. Looking at the case for n = 2, we can easily count all the different cases than can occur. We know that there are 4 matches and 12 mismatches, which leaves us with exactly 48 different ways to generate n =2 with inserts and deletes. The first way is to have consecutive deletes which exists a total of 16 different ones. The next one would be consecutive inserts, which exists a total of 16. Lastly, we consider the case of insert followed by a delete, which exists a total of 16 for n = 2. (We do not count the case of delete insert as that would make it not a canonical

pairwise alignment.) We would notice that we have an exact number of inserts to deletes in this case. Telling us that the number of deletes should roughly be the same to the number of inserts for size n. Based on the constants that would be the case. However, from out analytical way of determining the correct number, we would see that $\frac{3}{4}$ the canonical pairwise alignment would consist of deletes and inserts. However, that would be wrong, if we check n = 3 would start to notice that the number of matches and mismatches increases to about $\frac{1}{3}$ of all possible canonical pairwise alignments. Similarly, the number of deletes and inserts falls to about $\frac{2}{3}$. To do this one would need to spend a lot of time writing out all the different alignments and counting them.

Even after determining the correct proportions of matches, mismatches, deletes, and inserts in each canonical pairwise alignment, one still needs to know the average number of alignments needed for each n. In order to solve this analytically we need to estimate how many alignments are needed. While this can be done, since a match and mismatch take up size 2 and a delete and inserts take up size 1. It will take a lot of time. One method of doing it that was would be to generate all canonical pairwise alignments of size n and count the average alignments that way.

To conclude, while it is possible to determine the expected number of matches, mismatches, deletes, and inserts analytically, it would be time consuming to generate all the permutation for size n. Then determine the proportions of these permutation are matches and mismatches, (we know there are ¼ matches to mismatches) while the remaining out of 1 would be the proportions of deletes and inserts (we know there are the same number of inserts and deletes). Lastly, the need to count the average number alignments would take some time. As a result, randomly generated objects took a tedious task and used simple algorithms to generate the data needed to estimate the expected number of matches, mismatches, deletes, and inserts.